AS_rMATS

Matthew Taliaferro

6/25/2020

```
library(tidyverse)
library(reshape2)
library(RColorBrewer)
library(pheatmap)
```

Overview

This week we are using RNAseq to study alternative splicing. We talked last time about how this can be done using alignments of RNAseq reads to the genome. We aligned reads using `STAR`. Today, we will talk about how to take those alignments and turn them into quantifications of exon inclusion using `rMATS`. As is often the case with bioinformatic tools, `rMATS` is not the only tool that you can use to look at alternative splicing, but it has been around for a while and has been thoroughly tested.

#PSI values

In many scenarios, splicing is quantified using a metric called PSI (Percent Spliced In), often shown as the greek letter psi. PSI is a rather simple metric that asks what fraction of transcripts *contain* the exon or RNA sequence in question. PSI values therefore range from 0 (which would indicate that the exon is never included) to 1 (which would indicate that the exon is always included). PSI can be estimated by counting the number of reads that unambiguously support the inclusion of the exon and the number of reads that unambiguously support exclusion of the exon.

For skipped "cassette" exons, these reads are diagrammed below:

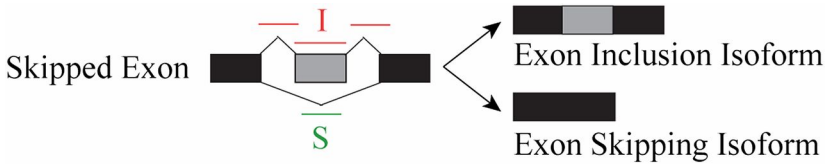


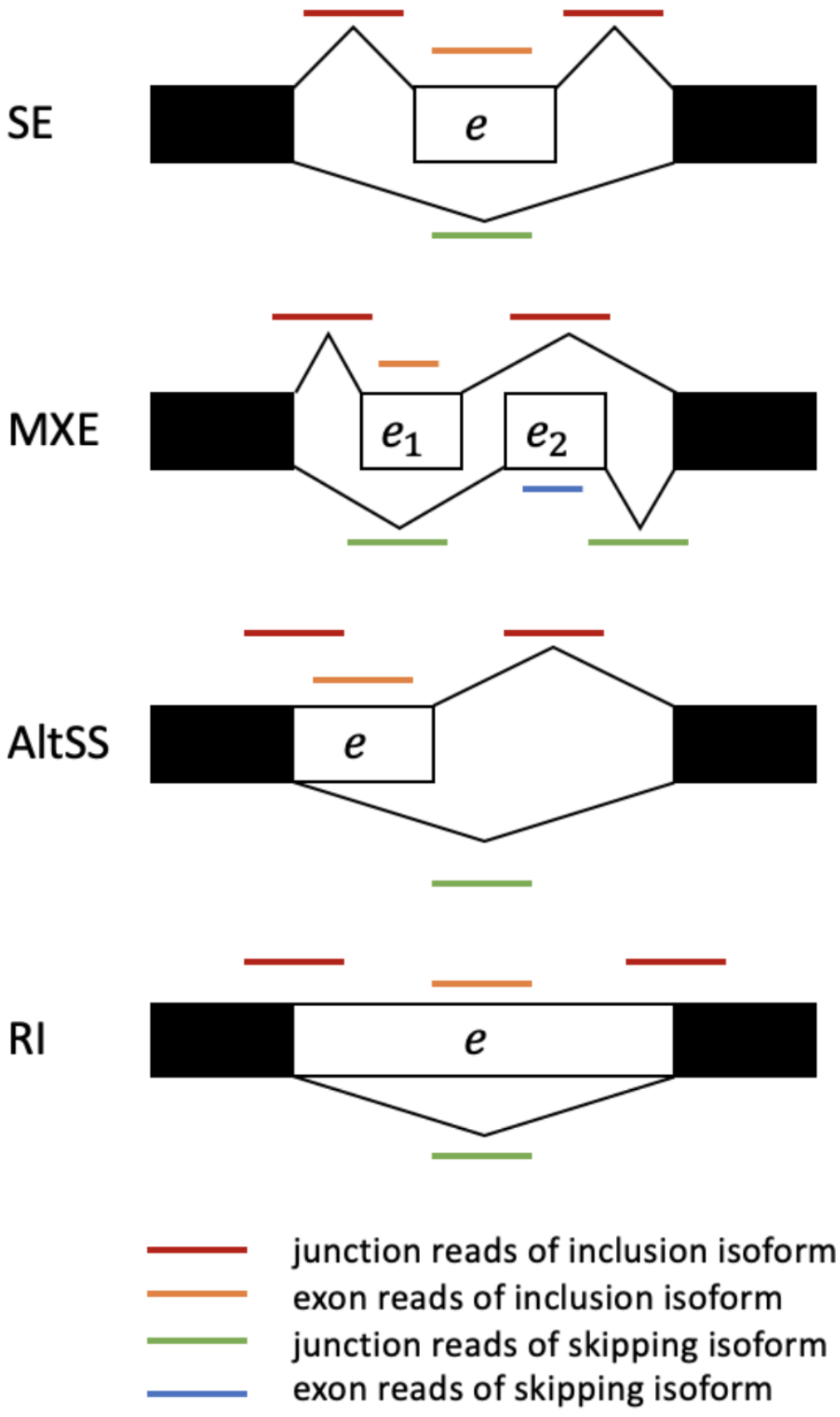
Figure 1: Shen et al, (2014). PNAS

In this diagram, the exon in gray can either be included or skipped to produced two different transcript isoforms. Reads in red (I for inclusion) unambiguously argue for the inclusion of the exon while reads in green (S for skipping) unambiguously argue for skipping of the exon. Keep in mind that the reads drawn over splice junctions indicate the the read spans the splice junction.

> Note: Red reads that do not cross a junction but lie totally within the gray exon are often used in splicing analysis, but do not formally unambiguously show exon inclusion. It could have been the case that the transcript that those reads originated from was just not spliced yet at all. If no introns have been removed yet, a read that does not cross any splice junctions could have come from a transcript in which the decision to retain or remove the gray exon has not yet been made. It is therefore safer to rely **only** on splice-junction spanning reads for splicing quantitation. The

downside of this is that you will lose read counts that came from non-junction reads. Fewer read counts means less statistical power.

As you may know, there are other types of alternative splicing besides skipped exons. Four other types are diagrammed below.


SE
e
MXE
e1
e2
AltSS
e
RI
e
junction reads of inclusion isoform
exon reads of inclusion isoform
junction reads of skipping isoform
exon reads of skipping isoform

In each case,

PSI is defined as the fraction of transcript in which the white sequence is included. For the purposes of this course,

Given that we have already determined where in the genome RNAseq reads came from using `STAR`, we need software that will take those locations and combine it with information about the locations of alternative exons to quantify the inclusion of each alternative exon. We will do that with `rMATS`.

#Running rMATS

To quantify alternative splicing, `rMATS` needs two things: the locations of the reads in the genome (bam files) and the locations of alternative exons in the genome (GTF annotation file).

> Note: You may remember that when we ran STAR, we used a different type of genome annotaiton file: GFF. GTFs and GFFs contain essentially the same information and it is possible to interconvert between the two. I chose to introduce you to GFFs because, to my mind, they are more intuitive to and readable by humans. STAR could handle both GTF and GFF formats. rMATS requires GTFs.

Here are the most relevant options when running `rMATS`. See the documentation here.

- –**b1** /path/to/b1.txt (path to a text file that contains paths to all BAM files for samples in condition 1)
- –**b2** /path/to/b2.txt (path to a text file that contains paths to all BAM files for samples in condition 2)
- –**gtf** /path/to/gtf (path to the gtf genome annotation)
- **-t** readtype (single or paired)
- –**readlength** readlength
- –**od** /path/to/output (output directory)

We won't run `rMATS` here because we would need multiple large bam files to do anything meaningful, and honestly, it's just copying things from the documentation and putting them into the command line. What we will do though, is look at the output produced by `rMATS`.

#Looking at rMATS output In the paper we are discussing, the authors were interested in the splicing regulatory activity of the RNA-binding protein RBFOX2. They sequenced RNA from cells that had been treated with either shRNA against RBFOX2 or a control, nontargeting shRNA. Each condition was performed in quadruplicate, meaning we have 4 replicates for each condition. I downloaded their data, aligned it against the mouse genome using `STAR`, and then quantified alternative splicing using `rMATS`.

`rMATS` produces many output files. Let's look at what it made.

```
ls -lh ../data/rMATS/RBFOX2kd/rMATSouts
```

```
## total 29632
## -rw-r--r--  1 mtaliaferro  staff   513K Jul 13 15:43 A3SS.MATS.JC.txt
## -rw-r--r--  1 mtaliaferro  staff   526K Jul 13 15:43 A3SS.MATS.JCEC.txt
## -rw-r--r--  1 mtaliaferro  staff   286K Jul 13 15:43 A5SS.MATS.JC.txt
## -rw-r--r--  1 mtaliaferro  staff   299K Jul 13 15:43 A5SS.MATS.JCEC.txt
## -rw-r--r--  1 mtaliaferro  staff   121K Jul 13 15:43 JC.raw.input.A3SS.txt
## -rw-r--r--  1 mtaliaferro  staff    67K Jul 13 15:43 JC.raw.input.A5SS.txt
## -rw-r--r--  1 mtaliaferro  staff    49K Jul 13 15:43 JC.raw.input.MXE.txt
## -rw-r--r--  1 mtaliaferro  staff   118K Jul 13 15:43 JC.raw.input.RI.txt
## -rw-r--r--  1 mtaliaferro  staff   602K Jul 13 15:43 JC.raw.input.SE.txt
## -rw-r--r--  1 mtaliaferro  staff   125K Jul 13 15:43 JCEC.raw.input.A3SS.txt
## -rw-r--r--  1 mtaliaferro  staff    71K Jul 13 15:43 JCEC.raw.input.A5SS.txt
## -rw-r--r--  1 mtaliaferro  staff    61K Jul 13 15:43 JCEC.raw.input.MXE.txt
## -rw-r--r--  1 mtaliaferro  staff   133K Jul 13 15:43 JCEC.raw.input.RI.txt
## -rw-r--r--  1 mtaliaferro  staff   687K Jul 13 15:43 JCEC.raw.input.SE.txt
## -rw-r--r--  1 mtaliaferro  staff   221K Jul 13 15:43 MXE.MATS.JC.txt
## -rw-r--r--  1 mtaliaferro  staff   252K Jul 13 15:43 MXE.MATS.JCEC.txt
## -rw-r--r--  1 mtaliaferro  staff   508K Jul 13 15:43 RI.MATS.JC.txt
```

```
## -rw-r--r--  1 mtaliaferro  staff   551K Jul 13 15:43 RI.MATS.JCEC.txt
## -rw-r--r--  1 mtaliaferro  staff   2.4M Jul 13 15:43 SE.MATS.JC.txt
## -rw-r--r--  1 mtaliaferro  staff   2.7M Jul 13 15:43 SE.MATS.JCEC.txt
## -rw-r--r--  1 mtaliaferro  staff   505K Jul 13 15:43 fromGTF.A3SS.txt
## -rw-r--r--  1 mtaliaferro  staff   290K Jul 13 15:43 fromGTF.A5SS.txt
## -rw-r--r--  1 mtaliaferro  staff   158K Jul 13 15:43 fromGTF.MXE.txt
## -rw-r--r--  1 mtaliaferro  staff   372K Jul 13 15:43 fromGTF.RI.txt
## -rw-r--r--  1 mtaliaferro  staff   2.3M Jul 13 15:43 fromGTF.SE.txt
## -rw-r--r--  1 mtaliaferro  staff   102B Jul 13 15:43 fromGTF.novelEvents.A3SS.txt
## -rw-r--r--  1 mtaliaferro  staff   102B Jul 13 15:43 fromGTF.novelEvents.A5SS.txt
## -rw-r--r--  1 mtaliaferro  staff    57K Jul 13 15:43 fromGTF.novelEvents.MXE.txt
## -rw-r--r--  1 mtaliaferro  staff    24K Jul 13 15:43 fromGTF.novelEvents.RI.txt
## -rw-r--r--  1 mtaliaferro  staff   609K Jul 13 15:43 fromGTF.novelEvents.SE.txt
```

You can see that there are many files here, and that each type of alternative splicing (A3SS, A5SS, MXE, RI, and SE) has files associated with it. Specifically each event type has 2 files: one that ends in 'JC.txt' and one that ends in 'JCEC.txt'. The 'JC.txt' files only use reads that cross splice junctions to quantify splicing (JC = junction counts) while the 'JCEC.txt' files use both junction reads *and* reads that map to the alternative exon (EC = exon counts). As we talked about above, I feel more comfortable only using junction counts, so we will use 'JC.txt' files.

Let's take a look at the output for the quantification of skipped exons.

```
head ../data/rMATS/RBFOX2kd/rMATSouts/SE.MATS.JC.txt
```

```
## ID   GeneID  geneSymbol  chr strand  exonStart_0base exonEnd upstreamES  upstreamEE  downstreamES
## 5    "ENSMUSG00000032298.13" "Neil1" chr9    -   57143757    57143819    57143450    57143613    5714
## 6    "ENSMUSG00000041346.11" "Wrap53"    chr11   -   69562121    69562259    69561757    69562019
## 9    "ENSMUSG00000062861.8"  "Zfp28" chr7    +   6392181 6392296 6390399 6390560 6393366 6396915 9
## 11   "ENSMUSG00000039936.18" "Pik3cd"    chr4    -   149656603   149656700   149656380   149656511
## 17   "ENSMUSG00000037706.17" "Cd81"  chr7    +   143065942   143066017   143064673   143065306   143
## 21   "ENSMUSG00000032187.16" "Smarca4"   chr9    +   21632804    21633054    21616168    21616400
## 22   "ENSMUSG00000032187.16" "Smarca4"   chr9    +   21677952    21678051    21677449    21677677
## 26   "ENSMUSG00000028849.17" "Map7d1"    chr4    -   126238521   126238632   126236895   126237278
## 27   "ENSMUSG00000028849.17" "Map7d1"    chr4    -   126238521   126238632   126236895   126237278
```

OK that's a mess, but hopefully you can see that this is just a text file that contains a tab delimited table, which means working with this in R should be straightforward. Column names are at the top. Let's go through some of the more important columns:

- **ID** A unique identifier for this event in this table. Useful for only that reason.
- **chr** chromosome
- **strand** strand that the transcript is on (+ or -)
- **exonStart_0base** the coordinate of the beginning of the alternative exon (using 0-based coordinates)
- **exonEnd** the coordinate of the end of the alternative exon
- **upstreamES** the coordinate of the beginning of the exon immediately upstream of the alternative exon
- **upstreamEE** the coordinate of the end of the exon immediately upstream of the alternative exon
- **downstreamES** the coordinate of the beginning of the exon immediately downstream of the alternative exon
- **downstreamEE** the coordinate of the end of the exon immediately downstream of the alternative exon

Notice that with these coordinates and the sequence of the genome, you could derive the sequences flanking each of these exons. That could be useful, perhaps, if you wanted to ask if there were short sequences (kmers) enriched near exons whose inclusion was sensitive to RBFOX2 loss versus exons whose inclusion was insensitive.

- **IJC_SAMPLE_X** the number of read counts that support inclusion of the exon is sample X (four numbers, one for each replicate, each separated by a comma)
- **SJC_SAMPLE_X** same thing, but for read counts that support the exclusion of the exon

These numbers could be useful for filtering events based on coverage. Say, for example, that we were looking at an event that when we combined IJC and SJC counts for each replicate we got something like 2,4,1,5. That would mean that in the replicates for this condition, we only had 2, 4, 1, and 5 reads that tell us anything about the status of this exon. That's pretty low, so I would argue that we really wouldn't want to consider this event at all since we don't have much confidence that we know anything about its inclusion.

- **PValue** The pvalue asking if the PSI values for this event between the two conditions is statistically significantly different
- **FDR** The p value, after it has been corrected for multiple hypothesis testing. This is the significance value you would want to filter on.
- **IncLevel1** PSI values for the replicates in condition 1 (in this case, condition 1 is RBFOX shRNA).
- **IncLevel2** PSI values for the replicates in condition 2 (in this case, condition 2 is Control shRNA).
- **IncLevelDifference** Difference in PSI values between conditions (Condition 1 - Condition 2).

Let's read this file in and work on it a bit.

```
psis <- read.table('../data/rMATS/RBFOX2kd/rMATSouts/SE.MATS.JC.txt', header = T) %>%
  #Get rid of columns we aren't really going to use.
  dplyr::select(., c('ID', 'geneSymbol', 'IJC_SAMPLE_1', 'SJC_SAMPLE_1', 'IJC_SAMPLE_2', 'SJC_SAMPLE_2'

head(psis)
```

```
##   ID geneSymbol    IJC_SAMPLE_1 SJC_SAMPLE_1    IJC_SAMPLE_2 SJC_SAMPLE_2 FDR
## 1  5      Neil1        12,6,9,5      0,0,0,0       10,8,14,0      0,1,2,0   1
## 2  6     Wrap53     44,34,40,22      3,0,0,0     29,13,19,15      0,0,0,0   1
## 3  9      Zfp28         3,0,0,4      0,0,0,0         2,0,0,0      3,0,0,0   1
## 4 11     Pik3cd        18,5,1,3      0,0,0,0        3,7,10,2      0,0,0,2   1
## 5 17       Cd81 348,210,261,209      0,0,0,0 231,166,135,231      0,0,2,0   1
## 6 21    Smarca4  136,107,135,78      0,0,1,0  107,132,95,122      0,0,0,0   1
##           IncLevel1         IncLevel2 IncLevelDifference
## 1   1.0,1.0,1.0,1.0  1.0,0.8,0.778,NA              0.141
## 2  0.88,1.0,1.0,1.0   1.0,1.0,1.0,1.0             -0.030
## 3     1.0,NA,NA,1.0     0.25,NA,NA,NA              0.750
## 4   1.0,1.0,1.0,1.0 1.0,1.0,1.0,0.333              0.167
## 5   1.0,1.0,1.0,1.0 1.0,1.0,0.971,1.0              0.007
## 6 1.0,1.0,0.985,1.0   1.0,1.0,1.0,1.0             -0.004
```

Based on our discussion above about read counts and confidence, I've decided that I only want to consider events where there are *at least* 20 informative reads that tell us about the inclusion of the exon (IJC + SJC) in **every replicate**. For example, for event '5' (gene Neil1) above, Sample 1 replicates have 12, 6, 9, and 5 reads while Sample 2 replicates have 10, 9, 16, and 0 reads. I would want to require that all of 12, 6, 9, 5, 10, 9, 16, and 0 are greater than 20 in order to worry about this event. Otherwise, I conclude that we don't have enough data to accurately conclude anything about this event.

```
psis <- psis %>%
  #Split the replicate read counts that are separated by commas into different columns
  separate(., col = IJC_SAMPLE_1, into = c('IJC_S1R1', 'IJC_S1R2', 'IJC_S1R3', 'IJC_S1R4'), sep = ',', 
  separate(., col = SJC_SAMPLE_1, into = c('SJC_S1R1', 'SJC_S1R2', 'SJC_S1R3', 'SJC_S1R4'), sep = ',', 
  separate(., col = IJC_SAMPLE_2, into = c('IJC_S2R1', 'IJC_S2R2', 'IJC_S2R3', 'IJC_S2R4'), sep = ',', 
```

```
  separate(., col = SJC_SAMPLE_2, into = c('SJC_S2R1', 'SJC_S2R2', 'SJC_S2R3', 'SJC_S2R4'), sep = ',', i

head(psis)
```

```
##   ID geneSymbol IJC_S1R1 IJC_S1R2 IJC_S1R3 IJC_S1R4 SJC_S1R1 SJC_S1R2 SJC_S1R3
## 1  5      Neil1       12        6        9        5        0        0        0
## 2  6     Wrap53       44       34       40       22        3        0        0
## 3  9      Zfp28        3        0        0        4        0        0        0
## 4 11     Pik3cd       18        5        1        3        0        0        0
## 5 17       Cd81      348      210      261      209        0        0        0
## 6 21    Smarca4      136      107      135       78        0        0        1
##   SJC_S1R4 IJC_S2R1 IJC_S2R2 IJC_S2R3 IJC_S2R4 SJC_S2R1 SJC_S2R2 SJC_S2R3
## 1        0       10        8       14        0        0        1        2
## 2        0       29       13       19       15        0        0        0
## 3        0        2        0        0        0        3        0        0
## 4        0        3        7       10        2        0        0        0
## 5        0      231      166      135      231        0        0        2
## 6        0      107      132       95      122        0        0        0
##   SJC_S2R4 FDR         IncLevel1         IncLevel2 IncLevelDifference
## 1        0   1   1.0,1.0,1.0,1.0  1.0,0.8,0.778,NA              0.141
## 2        0   1  0.88,1.0,1.0,1.0   1.0,1.0,1.0,1.0             -0.030
## 3        0   1     1.0,NA,NA,1.0     0.25,NA,NA,NA              0.750
## 4        2   1   1.0,1.0,1.0,1.0 1.0,1.0,1.0,0.333              0.167
## 5        0   1   1.0,1.0,1.0,1.0 1.0,1.0,0.971,1.0              0.007
## 6        0   1 1.0,1.0,0.985,1.0   1.0,1.0,1.0,1.0             -0.004
```

Now sum to get reads in each condition for each event and filter.

```
psis.filtered <- psis %>%
  mutate(., S1R1counts = IJC_S1R1 + SJC_S1R1) %>%
  mutate(., S1R2counts = IJC_S1R2 + SJC_S1R2) %>%
  mutate(., S1R3counts = IJC_S1R3 + SJC_S1R3) %>%
  mutate(., S1R4counts = IJC_S1R4 + SJC_S1R4) %>%
  mutate(., S2R1counts = IJC_S2R1 + SJC_S2R1) %>%
  mutate(., S2R2counts = IJC_S2R2 + SJC_S2R2) %>%
  mutate(., S2R3counts = IJC_S2R3 + SJC_S2R3) %>%
  mutate(., S2R4counts = IJC_S2R4 + SJC_S2R4) %>%
  #Filter on read counts
  filter(., S1R1counts >= 20 & S1R2counts >= 20 & S1R3counts >= 20 & S1R4counts >= 20 &
           S2R1counts >= 20 & S2R2counts >= 20 & S2R3counts >= 20 & S2R4counts >= 20)
```

OK cool. We now have a table that only contains exons that we are confident we can quantify.

Plot distribution of PSI values

People have observed that exons whose inclusion is not really regulated tend to have PSI values that are either very close to 0 or very close to 1. That is to say that these exons are pretty much always included or always skipped. On the other hand, exons whose inclusion is regulated tend to be a lot more flexible in their inclusion and therefore have PSI values that are more evenly spread between 0 and 1. Perhaps this makes sense. An exon whose inclusion is regulated should have it's inclusion be "moveable", i.e. not always stuck at 0 or 1.

Can we see this in our data?

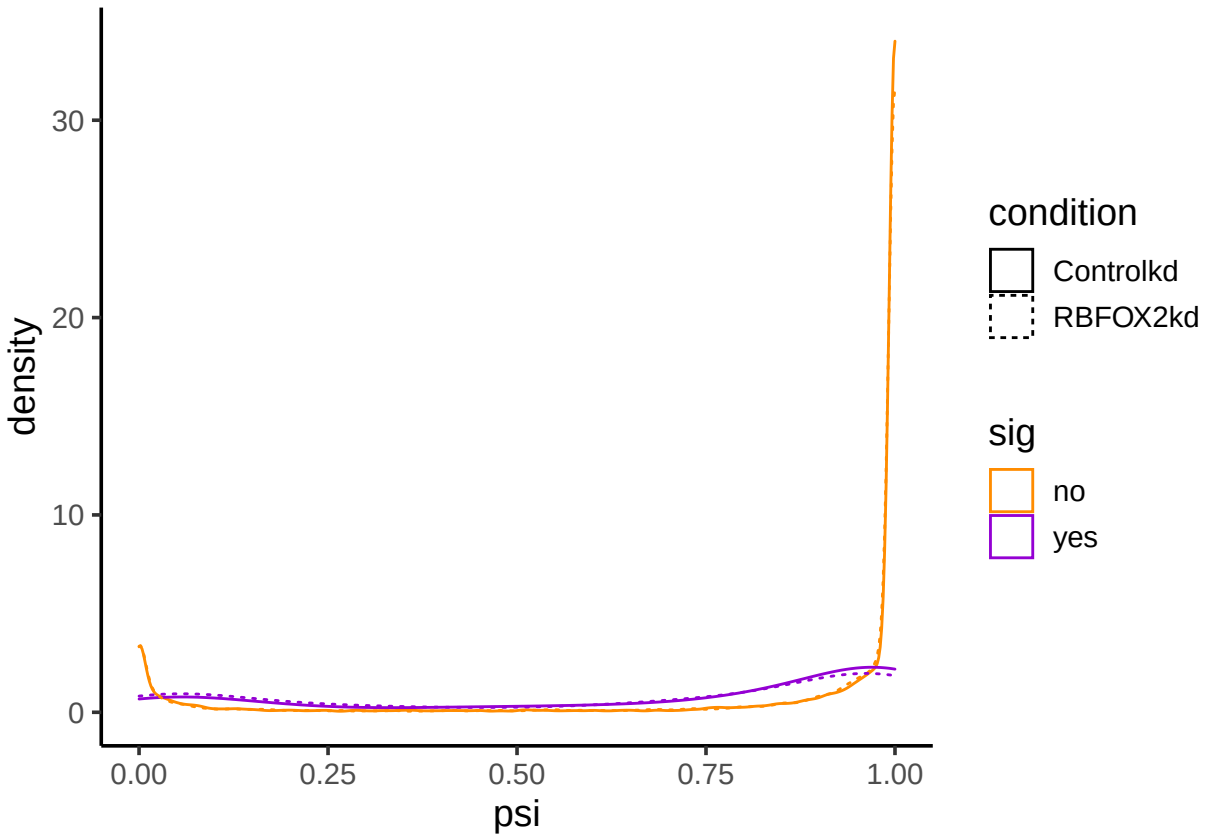
```
psis.filtered.psivalues <- psis.filtered %>%
  #Separate psi value replicates into individual columns
  separate(., col = IncLevel1, into = c('PSI_S1R1', 'PSI_S1R2', 'PSI_S1R3', 'PSI_S1R4'), sep = ',', reme
  separate(., col = IncLevel2, into = c('PSI_S2R1', 'PSI_S2R2', 'PSI_S2R3', 'PSI_S2R4'), sep = ',', reme
  #Select the columns we want
  dplyr::select(., c(contains('PSI'), FDR))

#Turn data from wide format into long format for plotting purposes
psis.filtered.psivalues.long <- gather(psis.filtered.psivalues, key = sample, value = psi, PSI_S1R1:PSI_
  #For each row, mark whether that replicate came from Condition 1 (RBFOXkd) or Condition2 (Controlkd)
  #We can tell that by asking if the substring 'S1' is somewhere in 'sample'
  mutate(., condition = ifelse(grepl('S1', sample), 'RBFOX2kd', 'Controlkd')) %>%
  #Make a column indicating whether the FDR in this row is significant
  mutate(., sig = ifelse(FDR < 0.05, 'yes', 'no'))

head(psis.filtered.psivalues.long)
```

```
##          FDR   sample   psi condition sig
## 1 1.00000000 PSI_S1R1 1.000  RBFOX2kd  no
## 2 1.00000000 PSI_S1R1 1.000  RBFOX2kd  no
## 3 1.00000000 PSI_S1R1 0.026  RBFOX2kd  no
## 4 0.09416153 PSI_S1R1 0.969  RBFOX2kd  no
## 5 0.17346273 PSI_S1R1 0.926  RBFOX2kd  no
## 6 1.00000000 PSI_S1R1 1.000  RBFOX2kd  no
```

```
#Plot
colors <- c('DarkOrange', 'DarkViolet')
ggplot(psis.filtered.psivalues.long, aes(x = psi, linetype = condition, color = sig)) + geom_density() +
  scale_color_manual(values = colors)
```



PCA of PSI values

Just as we did with gene expression values, we can monitor the quality of this data using principle components analysis. We would expect that replicates within a condition would be clustered next to each other in this analysis and that PC1, the prinicple component along which the majority of the variance lies, would separate the conditions.

```
#Make a matrix of psi values
psis.matrix <- dplyr::select(psis.filtered.psivalues, -FDR) %>%
  na.omit(.)

#Use prcomp() to derive principle component coordinants of PSI values
psi.pca <- prcomp(t(psis.matrix))

#Add annotations of the conditions to the samples
psi.pca.pc <- data.frame(psi.pca$x, sample = colnames(psis.matrix)) %>%
  mutate(., condition = ifelse(grepl('S1', sample), 'RBFOX2kd', 'Controlkd'))

#Get the amount of variances contained within PC1 and PC2
psi.pca.summary <- summary(psi.pca)$importance
pc1var = round(psi.pca.summary[2,1] * 100, 1)
pc2var = round(psi.pca.summary[2,2] * 100, 1)

#Plot PCA data
colors <- brewer.pal(2, 'Set1')
```

```
## Warning in brewer.pal(2, "Set1"): minimal value for n is 3, returning requested palette with 3 diffe
```

```
ggplot(psi.pca.pc, aes(x = PC1, y = PC2, shape = condition, color = condition)) + geom_point(size = 5)
  scale_color_manual(values = colors) + theme_classic(16) + xlab(paste('PC1,', pc1var, '% explained var
  ylab(paste('PC2,', pc2var, '% explained var.'))
```